1-2018

Variance-Optimal Offline and Streaming Stratified Random Sampling

Trong Duc Nguyen
Iowa State University, trong@iastate.edu

Ming-Hung Shih
Iowa State University

Divesh Srivastava
AT&T Labs–Research

Srikanta Tirthapura
Iowa State University, snt@iastate.edu

Bojian Xu
Eastern Washington University

Variance-Optimal Offline and Streaming Stratified Random Sampling

Abstract
Stratified random sampling (SRS) is a fundamental sampling technique that provides accurate estimates for aggregate queries using a small size sample, and has been used widely for approximate query processing. A key question in SRS is how to partition a target sample size among different strata. While Neyman's allocation provides a solution that minimizes the variance of an estimate using this sample, it works under the assumption that each stratum is abundant, i.e. has a large number of data points to choose from. This assumption may not hold in general: one or more strata may be bounded, and may not contain a large number of data points, even though the total data size may be large.

We first present VOILA, an offline method for allocating sample sizes to strata in a variance-optimal manner, even for the case when one or more strata may be bounded. We next consider SRS on streaming data that are continuously arriving. We show a lower bound, that any streaming algorithm for SRS must have (in the worst case) a variance that is $\Omega(r)$ away from the optimal, where r is the number of strata. We present S-VOILA, a practical streaming algorithm for SRS that is locally variance-optimal in its allocation of sample sizes to different strata. Both the offline and streaming algorithms are built on a method for reducing the size of a stratified random sample in a variance-optimal manner, which could be of independent interest. Our results from experiments on real and synthetic data show that that VOILA can have significantly smaller variance than Neyman's allocation (VOILA's variances are a factor of 1.4x-3000x smaller than that of Neyman allocation, with the same setting). The streaming algorithm S-VOIlA results in a variance that is typically close to VOILA, which was given the entire input beforehand.

Disciplines
Databases and Information Systems | Data Storage Systems | Electrical and Computer Engineering | Systems and Communications

Comments
This is a pre-print of the article Nguyen, Trong Duc, Ming-Hung Shih, Divesh Srivastava, Srikanta Tirthapura, and Bojian Xu. "Variance-Optimal Offline and Streaming Stratified Random Sampling." *arXiv preprint arXiv:1801.09039* (2018). Posted with permission.

Variance-Optimal Offline and Streaming Stratified Random Sampling

Trong Duc Nguyen
Iowa State University
trong@iastate.edu

Ming-Hung Shih
Iowa State University
mshih@iastate.edu

Divesh Srivastava
AT&T Labs–Research
divesh@research.att.com

Srikanta Tirthapura
Iowa State University
snt@iastate.edu

Bojian Xu
Eastern Washington University
bojianxu@ewu.edu

ABSTRACT

Stratified random sampling (SRS) is a fundamental sampling technique that provides accurate estimates for aggregate queries using a small size sample, and has been used widely for approximate query processing. A key question in SRS is how to partition a target sample size among different strata. While *Neyman allocation* provides a solution that minimizes the variance of an estimate using this sample, it works under the assumption that each stratum is abundant, i.e. has a large number of data points to choose from. This assumption may not hold in general: one or more strata may be bounded, and may not contain a large number of data points, even though the total data size may be large.

We first present `VOILA`, an offline method for allocating sample sizes to strata in a variance-optimal manner, even for the case when one or more strata may be bounded. We next consider SRS on streaming data that are continuously arriving. We show a lower bound, that any streaming algorithm for SRS must have (in the worst case) a variance that is $\Omega(r)$ factor away from the optimal, where r is the number of strata. We present `S-VOILA`, a practical streaming algorithm for SRS that is *locally variance-optimal* in its allocation of sample sizes to different strata. Both the offline and streaming algorithms are built on a method for reducing the size of a stratified random sample in a variance-optimal manner, which could be of independent interest. Our results from experiments on real and synthetic data show that that `VOILA` can have significantly smaller variance than Neyman allocation (`VOILA`'s variances are a factor of 1.4x-3000x smaller than that of Neyman allocation, with the same setting). The streaming algorithm `S-VOILA` results in a variance that is typically close to `VOILA`, which was given the entire input beforehand.

1. INTRODUCTION

Random sampling is a widely-used method for data analysis, and features prominently in the toolbox of virtually every approximate query processing system. The power of random sampling lies in its generality. For many important classes of queries, an approximate answer, whose error is small in a statistical sense, can be efficiently obtained through executing the query over an appropriately derived random sample. Sampling operators are part of all major database products, e.g. Oracle, Microsoft SQL Server, and IBM Db2; BlinkDB [3] is a system for approximate query answering that is based extensively on random sampling.

The simplest method for random sampling is *uniform random sampling*, where each element from the entire data (the "population") is chosen with the same probability. But uniform sampling may lead to a high variance in the estimate for aggregate queries, when there is a large variation in the contribution of different elements to the aggregate. An alternative method is *stratified random sampling* (SRS), where the population is partitioned into subgroups called "strata". Within each stratum, uniform random sampling is used to select a per-stratum sample. The different per-stratum samples are then combined to derive the "stratified random sample". In SRS, there is flexibility to emphasize some strata over others, through controlling the allocation of sample sizes to different strata; for instance, strata where there is higher variance can be given a larger allocation.

Suppose that there are r strata, numbered from 1 to r, and that the mean, variance, and number of items in the jth stratum are μ_j, σ_j^2, and n_j respectively. Suppose that the target sample size is k (total across all strata). We measure the quality of a stratified random sample through the variance in the estimate of the population mean, computed using this sample. In "uniform allocation", each stratum j gets an identical allocation of sample size of $k_j = k/r$. In "proportional allocation", a stratum is allocated a sample size proportional to the number of elements in it. Proportional allocation is more or less equivalent to uniform random sampling.[1] A commonly used allocation method that is believed to yield the smallest variance [25, 11] is "Neyman allocation", where stratum j gets an allocation proportional to $\sigma_j n_j$. Many sampling methods for approximate query processing, such as the ones used in [10, 3], are based on Neyman allocation.

Neyman allocation assumes that each stratum has abundant data, and it is possible to sample as many elements from the stratum as we need. However, in practice, some strata can be bounded, and may not contain sufficient elements to satisfy their allocation. To see this, consider that Neyman allocation assigns a sample of size $k_j = \frac{n_j \sigma_j}{\sum_{i=1}^{r} n_i \sigma_i} \cdot k$ to stratum j. A stratum j that has a large variance σ_j^2 relative to the other strata, yet a relatively small number of

[1] There is a minor difference. Both methods allocated a sample size proportional to the size of the stratum, but in case of uniform sampling, the sample size is proportional to the size of the stratum, in expectation, while in proportional allocation, the sample size is proportional to the size of the stratum.

points n_j, can receive a relatively large allocation k_j. It is possible that $k_j > n_j$, and the stratum may not have the required elements to satisfy its assigned allocation. We call such strata, which have a small number of elements, as "bounded" strata. For instance, in our experiments with the one-year-long OpenAQ dataset [27] on air quality measurements at different locations in the world, we found that after the first month, 11 out of 60 strata are bounded. This number drops to 3 bounded strata (out of 88) after 6 months and 1 (out of 177) after 12 months. For data with bounded strata, Neyman allocation is clearly no longer the variance-optimal method for allocating sample sizes, since it may assign a sample size greater than the number of points in the stratum.

Current methods for SRS also do not naturally extend to efficiently handle updates to data, since they were designed to only for static data. As a result, they are not effective for use on data streams, where data is continuously arriving. In this work, we consider the general problem of variance-optimal SRS in both the offline and streaming settings, when some of the strata may be bounded.

1.1 Our Contributions

– **Variance Optimal Stratified Random Sampling for Bounded Strata:** We present the first offline algorithm for variance-optimal SRS for data that may have bounded strata. Our algorithm `VOILA` (`V`ariance `O`pt`I`ma`L` `A`llocation) computes an allocation that has provably optimal variance among all possible allocations of sample sizes to different strata. While prior work assumes that there are no strata with small volumes of data, which is often violated in real data sets, our analysis makes no such assumptions. `VOILA` reduces to Neyman allocation in the case when every stratum is abundant.

– **Streaming Lower Bound:** We present a lower bound showing that any streaming algorithm for SRS that uses a memory of k records must have, in the worst case, a variance that is a factor of $\Omega(r)$ away from the variance of the optimal offline algorithm. Here, r denotes the number of strata. This lower bound is tight, since there exist streaming algorithms for SRS whose variance matches this bound in the worst case.

– **Practical Streaming Algorithm for SRS:** We present `S-VOILA`, a streaming algorithm for SRS that is locally optimal with respect to variance – upon receiving new elements, it (re-)allocates sample sizes among strata in such a way as to minimize the variance among all possible re-allocations. This can be viewed as the online, or dynamic counterpart of the optimization that led to `VOILA`, which is based on optimizing the variance using a static view of data. `S-VOILA` can also deal with the case when a minibatch of multiple data items is seen at a time, rather than only a single item at a time – re-allocations made by `S-VOILA` are locally optimal with respect to the entire minibatch, and are of higher quality for larger size minibatches than when a single element is seen at a time. In our experimental study, the variance of `S-VOILA` is very close to that of the offline algorithm `VOILA`. Since it can deal with minibatches of varying sizes, it is well-suited to real-world streams that may have bursty arrivals.

– **Variance Optimal Sample Size Reduction:** The algorithms for offline SRS (`VOILA`) and streaming SRS (`S-VOILA`) are both based on a technique for reducing the size of an existing stratified random sample down to a desired target size in such a way such that the variance of the estimator based on the final sample is as small as possible. This technique for sample size reduction may be of independent interest in other tasks such as sub-sampling from a given stratified random sample.

– **Experimental Evaluation:** We present a detailed experimental evaluation using real and synthetic data sets, considering both quality of sample and accuracy of query answers. Our experiments show that (a) `VOILA` can have significantly smaller variance than Neyman allocation, and (b) `S-VOILA` closely tracks the allocation as well as the variance of the optimal offline algorithm `VOILA`. As the size of the minibatch increases, the variance of the samples produced by `S-VOILA` decreases. A minibatch of size 100 provides a majority of the benefits, in our experiments on real-world data.

1.2 Related Work

Sampling has been widely used in approximate query processing on both static and streaming data [11, 22, 29, 28, 18]. The reservoir sampling [23, 30] method for maintaining a uniform random sample on a stream has been known for decades, and many variants have been considered, such as weight-based sampling [14, 8], stream sampling under insertion and deletion of elements [16], adaptive-size reservoir sampling [4], distinct sampling [17], sampling from a sliding window [7, 15, 9], and time-decayed sampling [13, 12].

SRS in the online setting [28] can be viewed as a type of weight based reservoir sampling where the weight of each stream element is changing dynamically, based on the statistics of the stratum the element belongs to. Since the weight of each stream element changes dynamically, even after it has been observed, prior work on weighted reservoir sampling [14] does not apply here, since it assumes that the weight of an element is known at the time of observation and does not change henceforth. [24] considered streaming SRS using population-based allocation, and [21] considered streaming SRS using power allocation, based on their prior work on adaptive reservoir sampling [4]. However, prior work does not consider provable guarantees on the resulting samples, or lower bounds for streaming SRS, like we do here.

SRS has been used widely in approximate query processing in database systems [2, 1, 6, 10, 20]. BlinkDB [3] is a recent system for parallel approximate query processing based on SRS, where a collection of multi-dimensional stratified samples are pre-selected from data, assuming a given query workload. All these however works however assume static data. With the emergence of data stream processing systems [5] and data stream warehousing systems [19], it is important to devise methods for streaming SRS with quality guarantees.

2. PRELIMINARIES AND OVERVIEW OF SOLUTION

2.1 Preliminaries

We consider the construction and maintenance of a stratified random sample of data that is either stored offline, or

arriving as a stream. Stratified sampling can be viewed as being composed of three parts – stratification, sample allocation, and sampling.

Stratification is a partitioning of the universe into a number of disjoint strata, such that the union of all strata equals the universe. Equivalently, it is the assignment of each data element to a unique stratum. Stratification is often a predefined function of one or more attributes of the data element. For example, the work of Chaudhuri et al. [10] stratifies a tuple within a database table based on the set of selection predicates in the query workload that the tuple satisfies. In the OpenAQ dataset, air quality data measurements can be stratified on the basis of geographic location and measurement type, so that tuples relevant to a query can typically be composed of the union of strata. Our work assumes that the universe has already been partitioned into strata, and that each tuple comes with a stratum identifier. This assumption fits the model assumed in [10, 3].

Our work deals with sample allocation, the task of partitioning the available memory budget of M samples among the strata. In the case of offline sampling, allocation needs to be done only once, after knowing the data in its entirety. In the case of streaming sampling, the allocation may need to be continuously re-adjusted as more data arrives, and the characteristics of different strata change.

The final sampling step chooses within each stratum, the assigned number of samples uniformly at random. In the case of offline stratified sampling, the sampling step can be performed in a second pass through the data after sample size allocation, using reservoir sampling on the subset of elements belonging to each stratum. In the case of streaming sampling, the sampling step is not as easy, since it needs to occur simultaneously with sample (re-)allocation, which may change the allocations to different strata over time.

Variance-Optimal Allocation. Given a data set, $R = \{v_1, v_2, \ldots, v_n\}$ of size n, whose elements are stratified into r strata, numbered $1, 2, \ldots, r$. For each $i = 1 \ldots r$, let S_i be a a uniform random sample of size s_i drawn without replacement from stratum i. Let $\mathbb{S} = \{S_1, S_2, \ldots, S_n\}$ denote the stratified random sample.

The sample mean of each per-stratum sample S_i of size s_i is: $\bar{y}_i = \frac{\sum_{v \in S_i} v}{s_i}$. Using the sample means of all strata, the population mean of R, μ_R can be estimated as: $\bar{y} = \frac{\sum_{i=1}^{r} n_i \bar{y}_i}{n}$. It can be shown that the expectation of $\bar{y}$ equals μ_R.

Given a memory budget of $M \leq n$ elements to store all the samples, so that $\sum_i s_i = M$, we address the question: What is the value of each s_i, the size of sample S_i, so as to minimize the variance of $\bar{y}$. The variance of $\bar{y}$ can be computed as follows (e.g. see Theorem 5.3 in [11]):

$$V = V(\bar{y}) = \frac{1}{n^2} \sum_{i=1}^{r} n_i(n_i - s_i)\frac{\sigma_i^2}{s_i} = \frac{1}{n^2} \sum_{i=1}^{r} \frac{n_i^2 \sigma_i^2}{s_i} - \frac{1}{n^2} \sum_{i=1}^{r} n_i \sigma_i^2. \quad (1)$$

We call the answer to this question as a *variance-optimal allocation* of sample sizes to different strata.

Neyman Allocation for Strata that are abundant. All previous studies on variance-optimal allocation assume that every stratum has a large volume of data, to fill its sample allocation. Under this assumption, Neyman allocation [25, 11] minimizes the variance V, and allocates a sample size for stratum i as $M \cdot (n_i\sigma_i) / \left(\sum_{j=1}^{r} n_j\sigma_j\right)$.

Given a collection of data elements R, we say a stratum i is *abundant*, if $n_i \geq M \cdot (n_i\sigma_i) / \left(\sum_{j=1}^{r} n_j\sigma_j\right)$. Otherwise, the stratum i is *bounded*. Clearly, Neyman allocation works only if each stratum is abundant, and does not work if one or more strata are bounded. We consider the general case of variance-optimal allocation where there may be bounded strata.

2.2 Solution Overview

We note that both offline and streaming SRS can be viewed as a problem of "sample size reduction" in a variance-optimal manner. With offline SRS, we can initially view the entire data as a (trivial) sample of zero variance, where the sample size is very large – this sample needs to be reduced to fit within the memory budget of M records. If this reduction is done in a manner that increases the variance by the smallest amount, the resulting sample is a variance-optimal sample of size M.

In the case of streaming SRS, the streaming algorithm maintains a current stratified random sample of size M. It also maintains the characteristics of each stratum, including the number of elements n_i and standard deviation σ_i, in a streaming manner using $O(1)$ space per stratum. When a set of new stream elements arrive, we can let the per-stratum reservoir sampling algorithms continue sampling as before. If the sample size increases due to this step, then we are again faced with a problem of sample size reduction – how can this be reduced to a sample of size M in a variance-optimal manner?

Based on the above observation, we first present a variance-optimal sample size reduction method in Section 3. We start with an algorithm for reducing the size of the sample by one element, followed by a general algorithm for reducing the size by $\beta \geq 1$ elements, and then present an improved algorithm with a faster runtime. The variance-optimal offline algorithm `VOILA` can be viewed as an application of sample size reduction – details are presented in Section 4. We present a tight lower bound for any streaming algorithm, followed by `S-VOILA`, an algorithm for streaming SRS in Section 5. Note that the streaming algorithm `S-VOILA` does not necessarily lead to a variance-optimal sample. Though the individual sample-size reduction steps performed during observation of the stream are locally optimal, the overall result may not be optimal. Further details are in Section 5. We present a detailed experimental study of our algorithms in Section 6.

3. VARIANCE-OPTIMAL SAMPLE SIZE REDUCTION

Suppose it is necessary to reduce an SRS of total size M to an SRS of total size $M' < M$. This will need to reduce the size of the samples of one or more strata in the SRS. Since the sample sizes are reduced, the variance of the resulting estimate will increase. We consider the task of *variance-optimal sample size reduction (`VOR`)*, i.e., how to partition the reduction in sample size among the different strata in such a way that the increase in the variance is minimized.

Consider Equation 1 for the variance of an estimate derived from the stratified random sample. Note that, for a given data set, a change in the sample sizes of different strata

s_i does not affect the parameters n, n_i, and σ_i. VOR can be formulated as the following non-linear program.

$$\text{Minimize} \sum_{i=1}^{r} \frac{n_i^2 \sigma_i^2}{s_i'} \tag{2}$$

subject to constraints:

$$0 \leq s_i' \leq s_i \text{ for each } i = 1, 2, \ldots, r \tag{3}$$

$$\sum_{i=1}^{r} s_i' = M' \tag{4}$$

We observe that, without Constraint 3, and if all strata are unbounded, the answer to the above optimization program is exactly the Neyman allocation under memory budget M'. However, we have to deal with the additional Constraint 3 and the possibility of a stratum being bounded, in an efficient manner. In the rest of this section, we present efficient approaches for computing the VOR.

3.1 Special Case: Reduction by One Element

We first present an efficient algorithm for the case where the size of a stratified random sample is reduced by one element. An example application of this case is in designing a streaming algorithm for SRS, when stream items arrive one at a time.

We introduce a terminology that we will use frequently in the rest of the paper. Given a memory budget M, the Neyman allocation size for stratum i is $M_i = M \cdot n_i\sigma_i / \sum_{j=1}^{r} n_j\sigma_j$. The task is to choose stratum i such that after reducing the sample size s_i by one element, the increase in variance V (Equation 1) is the smallest. Our solution is to choose stratum i such that the partial derivative of V with respect to s_i is the largest over all possible choices of i.

$$\frac{\partial V}{\partial s_i} = -\frac{n_i^2\sigma_i^2}{n^2}\frac{1}{s_i^2}.$$

We choose stratum ℓ where:

$$\begin{aligned} \ell &= \arg\max_i \left\{ \frac{\partial V}{\partial s_i} \,\middle|\, 1 \leq i \leq r \right\} = \arg\min_i \left\{ \frac{n_i\sigma_i}{s_i} \,\middle|\, 1 \leq i \leq r \right\} \\ &= \arg\max_i \left\{ \frac{s_i}{M_i'} \,\middle|\, 1 \leq i \leq r \right\}, \end{aligned} \tag{5}$$

where M_i' is the Neyman allocation size for stratum i under the new memory budget M'. Equation 5 is due to the fact that each M_i' is proportional to $n_i\sigma_i$. This gives the following lemma.

Lemma 1. *When required to reduce the size of an stratified random sample by one, the increase in variance of the estimated population mean is minimized if we reduce the size of S_ℓ by one, where $\ell = \arg\min_i \left\{ \frac{n_i\sigma_i}{s_i} \,\middle|\, 1 \leq i \leq r \right\}$.*

In the case where we have multiple choices for ℓ using Lemma 1, we choose the one where the current sample size s_ℓ is the largest. Algorithm SingleSSR for reducing the sample size by one is shown in Algorithm 1. It is straightforward to observe that the run time of the algorithm is $O(r)$.

3.2 Reduction by $\beta \geq 1$ Elements

We now consider the general case, where the sample size needs to be reduced by some number β, $1 \leq \beta \leq M$. A possible solution idea is to repeatedly apply the one-element reduction algorithm (Algorithm 1 from Section 3.1)

Algorithm 1: SingleSSR(): Variance-Optimal Sample Size Reduction by One

Output: The identifier of the stratum whose sample size shall be reduced by one.

1 **return** $\arg\min_i \left\{ \frac{n_i\sigma_i}{s_i} \,\middle|\, 1 \leq i \leq r \right\}$

β times. Each iteration, a single element is chosen from a stratum such that the overall variance increases by the smallest amount. However, this greedy approach may not yield a sample with the smallest resulting variance. On the other hand, an exhaustive search of all possible evictions is not feasible either, since the number of possible ways to partition a reduction of size β among r strata is $\binom{\beta+r-1}{r}$, which is exponential in r and a high degree polynomial in β, which can be very large. We now present efficient approaches to VOR. We first present a recursive algorithm, followed by a faster iterative algorithm. Before presenting the algorithm, we present the following useful characterization of a variance-optimal reduction.

Definition 1. *We say that stratum i is* oversized *under memory budget M, if its allocated sample size $s_i > M_i$. Otherwise, we say that stratum i is* not oversized.

Lemma 2. *Suppose that E is the set of β elements that are to be evicted from a stratified random sample such that the variance V after eviction is the smallest possible. Then, each element in E must be from a stratum whose current sample size is oversized under the new memory budget $M' = M - \beta$.*

Proof. We use proof by contradiction. Suppose one of the evicted elements, is deleted from a sample S_α such that the sample size s_α is not oversized under the new memory budget. Because the order of the eviction of the β elements does not impact the final variance, suppose that element e is evicted after the other $\beta - 1$ evictions have happened. Let s_α denote the size of sample S_α at the moment t right after the first $\beta - 1$ evictions and before evicting e. The increase in variance caused by evicting an element from S_α is

$$\begin{aligned} \Delta &= \frac{1}{n^2}\left(\frac{n_\alpha^2\sigma_\alpha^2}{s_\alpha(s_\alpha - 1)}\right) = \left(\frac{\sum_{i=1}^{r} n_i\sigma_i}{nM'}\right)^2 \frac{{M'}_\alpha^2}{s_\alpha(s_\alpha - 1)} \\ &> \left(\frac{\sum_{i=1}^{r} n_i\sigma_i}{nM'}\right)^2 \end{aligned}$$

where ${M'}_\alpha = M' \frac{n_\alpha\sigma_\alpha}{\sum_{i=1}^{r} n_i\sigma_i}$ is the Neyman allocation for stratum α under memory budget M'. The last inequality is due to the fact that S_α is not oversized under budget M' at time t, i.e., $s_\alpha \leq M_\alpha'$.

Note that an oversized sample must exist at time t, since there are a total of $M' + 1$ elements in the stratified random sample at time t, and the memory target is M'. Instead of evicting e, if we choose to evict another element e' from an oversized sample $S_{\alpha'}$, the resulting increase in variance will be:

$$\begin{aligned} \Delta' &= \frac{1}{n^2}\left(\frac{n_{\alpha'}^2\sigma_{\alpha'}^2}{s_{\alpha'}(s_{\alpha'} - 1)}\right) = \left(\frac{\sum_{i=1}^{r} n_i\sigma_i}{nM'}\right)^2 \frac{{M'}_{\alpha'}^2}{s_{\alpha'}(s_{\alpha'} - 1)} \\ &< \left(\frac{\sum_{i=1}^{r} n_i\sigma_i}{nM'}\right)^2 \end{aligned}$$

Algorithm 2: SSR($\mathcal{A}, M, \mathcal{L}$): Variance-Optimal Sample Size Reduction

Input: $\mathcal{A}$ is the set of strata under consideration. M is the target for the total sample size of all strata in $\mathcal{A}$.

Output: For $i \in \mathcal{A}$, $\mathcal{L}[i]$ is set to the final size of sample for stratum i, such that the increase of the variance V is minimized.

```
1  O ← ∅   // oversized samples
2  for j ∈ A do
3      M_j ← M · n_j σ_j / Σ_{t∈A} n_t σ_t // Neyman allocation
           if memory M divided among A
4      if s_j > M_j then O ← O ∪ {j}
5      else L[j] ← s_j // Keep current allocation
6
7  if O = A then
       /* All samples are oversized.  Recursion
          stops.                                  */
8      for j ∈ A do L[j] ← M_j
9  else
       /* Recurse on strata in O, under remaining
          memory budget.                          */
10     SSR(O, M − Σ_{j∈A−O} s_j, L)
```

where $M'_{\alpha'} = M' \frac{n_{\alpha'}\sigma_{\alpha'}}{\sum_{i=1}^{r} n_i \sigma_i}$ is the Neyman allocation for stratum α' under memory budget M'. The last inequality is due to the fact that $S_{\alpha'}$ is oversized under budget M' at time t, i.e., $s_{\alpha'} > M'_{\alpha'}$. Because $\Delta' < \Delta$, at time t, evicting e' from $S_{\alpha'}$ leads to a lower variance than evicting e from S_α. This is a contradiction to the assumption that evicting e leads to the smallest variance, and completes the proof. □

Lemma 2 implies that it is only necessary to reduce the size of the samples that are oversized under the target memory budget M'. Samples that are not oversized can be given their current allocation, even under the new memory target M'. Our algorithm based on this observation first allocates sizes to the samples that are not oversized. The remaining memory now needs to be allocated among the oversized samples. Since this can again be viewed as a sample size reduction problem, while focusing on a smaller set of (oversized) samples, this is accomplished using a recursive call under a reduced memory budget; See Lemma 3 for a formal statement of this idea. The base case for this recursion is when all samples under consideration are oversized. In this case, we simply use the Neyman allocation to each stratum, under the reduced memory budget M' (Observation 1). Our algorithm SSR is shown in Algorithm 2.

Let $\mathbb{S} = \{S_1, S_2, \ldots, S_r\}$ be the current stratified random sample. Let $\mathcal{A}$ denote the set of all strata under consideration, initialized to $\{1, 2, \ldots, r\}$. Let $\mathcal{O}$ denote the set of oversized samples, under target memory budget for $\mathbb{S}$, and $\mathcal{U} = \mathbb{S} - \mathcal{O}$ denote the collection of samples that are not oversized. When the context is clear, we use $\mathcal{O}$, $\mathcal{U}$, and $\mathcal{A}$ to refer to the set of stratum identifiers as well as the set of samples corresponding to these identifiers.

Lemma 3. *A variance-optimal eviction of β elements from $\mathbb{S}$ under memory budget M' requires a variance-optimal eviction of β elements from $\mathcal{O}$ under memory budget $M' - \sum_{j\in\mathcal{U}} s_j$.*

Proof. Recall that s'_i denotes the final size of sample S_i after β elements are evicted. Referring to the variance V from Equation 1, we know a variance-optimal sample size reduction of β elements from $\mathbb{S}$ under memory budget M' requires to minimize

$$\sum_{i\in\mathcal{A}} \frac{n_i^2\sigma_i^2}{s'_i} - \sum_{\in\mathcal{A}} \frac{n_i^2\sigma_i^2}{s_i} \tag{6}$$

By Lemma 2, we know $s_i = s'_i$ for all $i \in \mathcal{U}$. Hence, minimizing Formula 6 is equivalent to minimizing

$$\sum_{i=\mathcal{O}} \frac{n_i^2\sigma_i^2}{s'_i} - \sum_{i\in\mathcal{O}} \frac{n_i^2\sigma_i^2}{s_i} \tag{7}$$

The minimization of Formula 7 is exactly the result obtained from a variance-optimal sample size reduction of β elements from oversized samples under the new memory budget $M' - \sum_{i\in\mathcal{U}} s_i$. □

Observation 1. *In the case every sample in the stratified random sample is oversized under target memory M', i.e., $\mathbb{S} = \mathcal{O}$, the variance-optimal reduction is to reduce the size of each sample $S_i \in \mathbb{S}$ to its Neyman allocation M'_i under the new memory budget M'.*

Theorem 1. *Algorithm 2 (SSR) finds a variance-optimal reduction of the stratified random sample $\mathcal{A}$ under new memory budget M. The worst-case time of SSR is $O(r^2)$, where r is the number of strata.*

Proof. Correctness follows from Lemmas 2–3 and Observation 1. The worst-case time happens when each recursive call sees only one stratum that is not oversized. In such a case, the total time of all recursions of SSR on a stratified random sample across r strata is: $O(r+(r-1)+\ldots+1) = O(r^2)$. □

Although SSR takes $O(r^2)$ time in the worst case, its time complexity tends to be much better in practice. If the number of samples that are not oversized contributes at least a certain percentage of the total number of samples being considered in every recursion, its overall time cost will be $O(r)$.

We also present an iterative algorithm, FastSSR, for sample size reduction that has a better computational cost, of $O(r \log r)$. FastSSR shares the same algorithmic foundation as SSR, but uses a faster method to find all the samples that are not oversized, leading to an overall faster algorithm for variance-optimal sample size reduction. Due to space constraints, we only state the properties of FastSSR and present details of the algorithm and its proof in the full version of the paper [26].

Theorem 2. *There is an algorithm FastSSR for variance-optimal sample size reduction on r strata, whose worst-case time complexity is $O(r \log r)$.*

4. VOILA: VARIANCE-OPTIMAL OFFLINE SRS FOR BOUNDED STRATA

In this section, we present an algorithm for computing the variance-optimal allocation of sample sizes in the case when one or more strata may be bounded. This allocation generalizes and extends the classic Neyman allocation, which is variance-optimal for the case when there is abundant data

within each stratum. The actual sampling step is straightforward for the offline algorithm – once the allocation of sample sizes is determined, the samples can be chosen in a second pass through the data, using reservoir sampling within each stratum. Hence, in the rest of this section, we focus on determining the allocation.

Consider a static data set R of n elements across r strata, where stratum i has n_i elements, and has standard deviation σ_i. *Given a memory budget of M elements, how can this be allocated among the strata such that the variance of an estimate of the population mean using the stratified sample is minimized? It is possible that some strata may have very few elements in them, but the total number of elements across all strata is at least M.* Note that simply using Neyman allocation which allocates samples to stratum i in proportion to $n_i\sigma_i$ may not be optimal. The reason is that this may grant a stratum a large allocation due to its high variance, but there may not be enough data in the stratum to fill this allocation. We present `VOILA` (**V**ariance-**O**pt**I**ma**L** **A**llocation), an efficient algorithm for the above question. `VOILA` is a generalization of the classic Neyman allocation – in the case when every stratum has abundant data, it reduces to Neyman allocation.

The idea is as follows. Consider the expression for the variance V in Equation 1. Since parameters n, r, n_i, and σ_i, are constants that cannot be affected by this decision, minimizing V only requires the minimization of $\sum_{i=1}^{r}(n_i^2\sigma_i^2/s_i)$. This leads to the following optimization problem.

$$\text{Minimize } \sum_{i=1}^{r} \frac{n_i^2\sigma_i^2}{s_i} \tag{8}$$

subject to constraints:

$$0 \le s_i \le n_i, \text{ for each } i = 1, 2, \ldots, r \tag{9}$$

$$\sum_{i=1}^{r} s_i = M \tag{10}$$

The similarity in structure between this optimization problem and `VOR`, formulated in (2)–(4), leads us to consider the following **hypothetical two-step process** that reduces variance-optimal offline SRS to variance-optimal sample size reduction.

Step 1: Suppose we start with a memory budget of n. Then, we will just save the whole data set in the stratified random sample, and thus each sample size $s_i = n_i$. By doing so, the variance V is minimized, since $V = 0$ (Equation 1).

Step 2: Given the stratified random sample from Step 1, we reduce the memory budget from n to M such that the resulting variance is the smallest. This can be done using variance-optimal sample size reduction, by calling `SSR` or `FastSSR` with target sample size M.

`VOILA` (Algorithm 3) simulates this process. The algorithm only records the sample sizes of the strata in array $\mathcal{L}$, without creating the actual samples. The actual sample from stratum i is created by choosing $\mathcal{L}[i]$ random elements from stratum i, using any method for offline uniform random sampling without replacement.

Theorem 3. *Given a data set $\mathcal{R}$ with r strata, and a memory budget M,* `VOILA` *(Algorithm 3) returns in $\mathcal{L}$ the sample size of each stratum in a variance-optimal stratified random sample. The worst-case time cost of* `VOILA` *is $O(r \log r)$.*

Algorithm 3: `VOILA` (M): Variance-optimal stratified random sampling for bounded data

```
Input: M is the memory target
for i = 1 ... r do
    s_i ← n_i  // assume total available memory of n
L ← FastSSR(M)
return L   /* L[i] ≤ n_i is the desired size of S_i
   in a variance-optimal stratified random sample.
   The actual samples can be constructed by
   choosing, for each i, a random sample of size
   L[i] from R_i.  */
```

Proof. The correctness follows from the correctness of Theorem 2, since the final sample is the sample of the smallest variance that one could obtain by reducing the initial sample (with zero variance) down to a target memory of size M. The run time is dominated by the call to `FastSSR`, whose time complexity is $O(r \log r)$. □

5. STREAMING SRS

We now consider the maintenance of an SRS from a data stream, whose elements are arriving continuously.

5.1 A Lower Bound for Streaming SRS

Given a data stream $\mathcal{R}$ across r strata, let V^* denote the sample variance of the stratified random sample created by `VOILA`, using a memory budget of M. Because `VOILA` is variance optimal, V^* is the smallest variance that we can get from any stratified random sample of $\mathcal{R}$ under the memory budget M. While `VOILA` is not a streaming algorithm, V^* is a lower bound on the variance that a streaming algorithm can achieve, under memory budget M.

Let V denote the sample variance of an SRS of $\mathcal{R}$ using the same memory budget M. We say V is an approximation of V^* with a multiplicative error of α, for some constant $\alpha \ge 1$, if: (1) the sample within each stratum i is chosen uniformly at random without replacement from stratum i. (2) $V \le \alpha \cdot V^*$.

Theorem 4. *Any streaming algorithm for maintaining an SRS over a stream with r strata using a memory of M records must, in the worst case, have a multiplicative error $\Omega(r)$ when compared with the optimal variance that can be achieved by a stratified random sample using memory of M records.*

We present the proof of this result in the full version [26] of the paper, due to space constraints. The idea in the proof is to construct an input stream with r strata where the variance of all strata are the same until a certain point, where the variance of a single stratum increases to a high value – a variance-optimal SRS will respond by increasing the allocation to this stratum. However, a streaming algorithm is unable to do so quickly, since it is in general unable to collect enough samples to satisfy the increased allocation to this stratum. Though a streaming algorithm is able to compute the variance-optimal allocation to different strata in an online manner, it cannot actually maintain these samples using limited memory.

We also note that the above lower bound is tight, since the simple uniform allocation, which allocates M/r memory to each of the r strata that have been observed so far, has a variance which is within a multiplicative factor of r of the optimal. However, we see that the policy of uniform allocation performs poorly in practice, since it does not distinguish between different strata, whether based on volume or variance.

5.2 S-VOILA: A Practical Algorithm for Streaming SRS

We now present S-VOILA, a streaming algorithm for stratified random sampling. Choices made by S-VOILA are "locally optimal" in the following sense: when new stream elements arrive, the decision of whether or not to select these elements (which will make it necessary to discard sampled elements from other strata) is made in a way that minimizes the variance of the estimate from resulting sample. S-VOILA can be viewed as an online version of VOILA, which constructs an SRS with minimal variance using a multi-pass algorithm through the entire data.

Let $\mathcal{R}$ denote the stream so far, and $\mathcal{R}_i$ the substream of elements belonging to stratum i. Within a single stratum, any algorithm for SRS needs to maintain a uniform random sample of all data seen so far. In streaming SRS, the memory s_i allocated to a stratum i may change with time, depending on the data arriving within this stratum, and other strata. One issue for a streaming algorithm is to maintain a uniform random sample within stratum i when s_i is changing. A decrease in the allocation s_i can be handled easily, through discarding randomly chosen elements from the current sample S_i until the desired sample size is reached. What if we need to increase the allocation to stratum i? If we simply start sampling new elements according to the higher allocation to stratum i, then recent elements in the stream will be favored over the older ones, and the sample within stratum i is no longer uniformly chosen. In order to ensure that S_i is always chosen uniformly at random from $\mathcal{R}_i$, newly arriving elements in $\mathcal{R}_i$ need to be held to the same sampling threshold as older elements, even if the allotted sample size s_i increases.

S-VOILA maintains sample S_i as follows. An arriving element from $\mathcal{R}_i$ is assigned a random "key" drawn uniformly from the interval $(0, 1)$. The algorithm maintains the following invariant: *S_i is the set of s_i elements with the smallest keys among all elements so far in $\mathcal{R}_i$.* Note that this means that if we desire to increase the allocation to stratum i, then this may not be accomplished immediately, since a newly arriving element in $\mathcal{R}_i$ may not be assigned a key that meets this sampling threshold. Instead, the algorithm has to wait until it receives an element in $\mathcal{R}_i$ whose assigned key is small enough. In order to ensure the above invariant, the algorithm maintains, for each stratum i, a variable d_i that tracks the smallest key of an element in $\mathcal{R}_i$ that is not currently included in S_i. If an arriving element in $\mathcal{R}_i$ has a key that is smaller than or equal to d_i, it is included within S_i; otherwise, it is not.

Algorithm 4 presents the initialization of S-VOILA, which simply loads the first M stream elements into the memory budget and divides them into r samples $S_1, S_2, \ldots, S_r$, and initializes state. As new elements arrive, they change the frequency and the variance of a stratum and may lead to changes in the desired allocation of samples to strata.

Algorithm 4: S-VOILA: Initialization

Input: Input parameters: M is the total sample size, r is the number of strata.

// S_i is the sample for stratum i, and $\mathcal{R}_i$ is the substream of elements from Stratum i

1 Load the first M stream elements in memory. Assign each element e in memory a key d chosen uniformly at random from $(0, 1)$.

2 Divide the M elements into r groups, $S_1, S_2, \ldots, S_r$, such that S_i consists of (e, d) tuples from stratum i, where e is the element, d is the key of the element.

3 **for** $i = 1 \ldots r$ **do**

4 Compute n_i, the number of elements in $\mathcal{R}_i$, and σ_i, the standard deviation of elements in $\mathcal{R}_i$ (so far)

5 $d_i \leftarrow 1$ // d_i is the smallest key among all elements in $\mathcal{R}_i$ not selected in S_i.

While it is possible to recompute the variance-optimal allocation, it is not possible to sample additional elements into strata as necessary, since we do not have the ability to look at all the data seen so far. However, our algorithm *locally optimizes* the variance through carefully selecting the strata from which samples will be discarded to make way for one or more incoming sampled elements.

S-VOILA supports the insertion of a minibatch of any size $b \geq 1$, where the value of b is even allowed to be dynamic during the execution of S-VOILA. When users fix $b = 1$, S-VOILA becomes streaming algorithm that handles one element at a time. As the value b increases, we can expect S-VOILA to have a better variance, since its optimization decisions are based on greater amount of data. Algorithm 5 presents the algorithm for maintaining the stratified random sample when a new minibatch of multiple elements arrives. Lines 2–7 make one pass through the minibatch to update the statistics of each stratum and store the selected elements into the sample. If $\beta > 0$ elements from the minibatch get selected into the sample, in order to balance the memory budget at M, we will need to evict β elements from the stratified random sample– this is accomplished using the variance-optimal sample size reduction technique from Section 3. For the special case where we only need to evict one element, we can use the faster algorithm SingleSSR (Lines 8–11); otherwise, FastSSR is used (Lines 12–17).

Theorem 5. *For each $i = 1, 2, \ldots, r$ sample S_i maintained by S-VOILA (Algorithm 5) is selected uniformly at random without replacement from stratum R_i.*

Proof. First, note that each S_i is selected from $\mathcal{R}_i$ without replacement, because each element of $\mathcal{R}_i$ is selected into S_i no more than once. Next, we prove the uniformity of S_i. In case $|S_i| = n_i$, all elements of $\mathcal{R}_i$ are in S_i. In case $|S_i| < n_i$, S_i contains the $|S_i|$ elements with the smallest keys from stratum $\mathcal{R}_i$, because: (1) Anytime an element is discarded from S_i, it is the element of the largest key in the sample. (2) If another element with key d enters later, it cannot be inserted into S_i unless d is smaller than or equal to all other keys discarded so far. Because the keys of elements are assigned randomly, each element has a chance of $|S_i|/n_i$ to be selected into S_i. Therefore, S_i is a uniform random sample from $\mathcal{R}_i$ without replacement. □

Algorithm 5: `S-VOILA`: Process a new minibatch B of b elements. Note: the value of b does not have to be fixed and is even allowed to dynamically change over the run of `S-VOILA`.

```
1  β ← 0;      // #selected elements in the minibatch
2  for each element e ∈ B do
3      Let α denote the stratum of e
4      Update n_α and σ_α
5      Assign a random key d ∈ (0,1) to element e;
6      if d ≤ d_α then          // element e is selected
7          S_α ← {e} ⋃ S_α; β ← β + 1;
   /* Variance-optimal eviction of β elements
      under memory budget M                          */
8  if β = 1 then     // faster for evicting 1 element
9      ℓ ← SingleSSR();
10     Delete one element of largest key from S_ℓ;
11     d_ℓ ← smallest key discarded from S_ℓ;
12 else if β > 1 then
13     L ← FastSSR(M);
14     for i = 1 ... r do    // Actual element evictions
15         if L[i] < s_i then
16             Delete s_i − L[i] elements of largest keys
                 from S_i;
17             d_i ← smallest key discarded from S_i;
```

Theorem 6. *If the minibatch size $b = 1$, then the worst-case time cost of `S-VOILA` for processing an element is $O(r)$. Further, the expected time for processing an element belonging to stratum α is $O(1 + r \cdot s_\alpha/n_\alpha)$, which is $O(1)$ when $r \cdot s_\alpha = O(n_\alpha)$.*

If $b > 1$, then the worst-case time cost of `S-VOILA` for processing a minibatch is $O(r \log r + b)$. The per-element amortized time cost is $O(1)$ when $b = \Omega(r \log r)$.

Proof. $b = 1$: The worst case happens when the single new element from belonging to stratum α gets selected into S_α. In that case, we need to reduce the stratified random sample size by one via `SingleSSR`, which takes $O(r)$ time. The probability that the new element is selected into S_α is equal to s_α/n_α, so the expected time follows.

$b > 1$: The time cost for Lines 2–7 is $O(b)$. The time cost for Lines 8–17 is $O(r \log r + \beta)$. So the total time cost is $O(b) + O(r \log r + \beta) = O(r \log r + b)$. □

We can expect `S-VOILA` to have an amortized per-item processing time of $O(1)$ in many circumstances.

For the case where $b = 1$: After observing enough stream elements from stratum α, such that $r \cdot s_\alpha = O(n_\alpha)$, which expects to be the case quickly in massive data stream processing, the expected processing time of an element becomes $O(1)$. Even if stratum α has a very low frequency, the expected time cost by `S-VOILA` for processsing a minibatch of size one is still expected to be $O(1)$, because elements from an infrequent stratum α will be unlikely to appear in the minibatch.

For the case where $b > 1$: The per-element amortized time cost of `S-VOILA` is $O(1)$, when the minibatch size $b = \Omega(r \log r)$.

6. EXPERIMENTAL EVALUATION

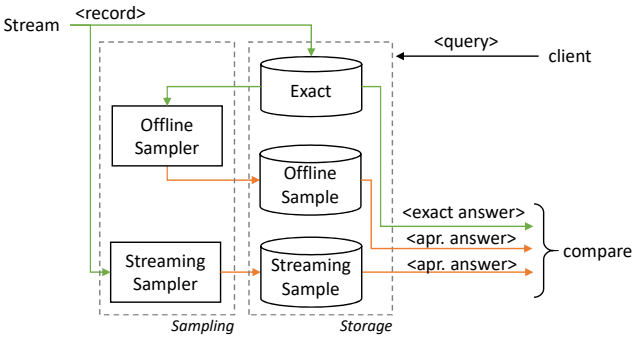


Figure 1: Setup for Evaluating Performance of Samples.

The algorithms developed are evaluated on real-world as well as synthetic data.

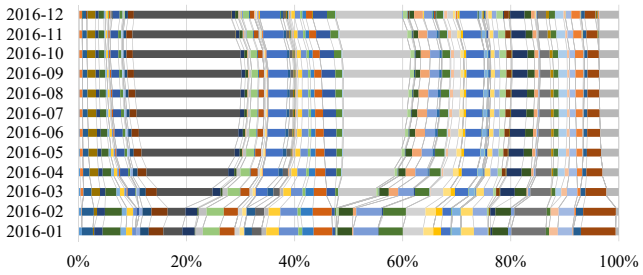
The setup used is shown in Figure 1, where input is a stream of records from a data source, and is fed into a stream sampling module. The stream sampler continuously maintains an SRS of data (stored in memory) from the stream observed so far. The stream sampler can process data in only a single pass, in constructing the sample. If a data element that has already arrived and is not stored within the memory, the stream sampler is not able to access it anymore. There is also an offline sampler module that has access to all data received so far, in computing the stratified random sample. When a sample is desired, the offline sampler can perform a multi-pass computation through all the data received so far, and compute a stratified random sample.

We evaluate the algorithms in two ways. The first is a direct evaluation of the quality of the samples, with respect to the variance of an estimate of the population mean obtained using the samples. The second is through the accuracy of approximate query processing using the samples.

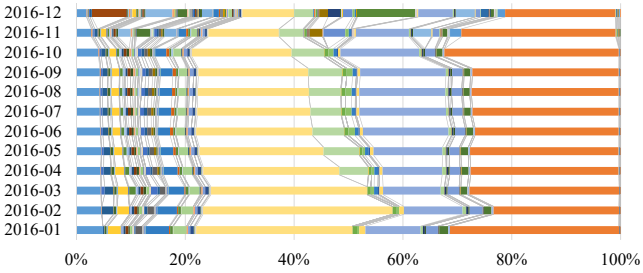
6.1 Sampling Methods

We implemented the following stream sampling methods: `S-VOILA` with different minibatch sizes, reservoir sampling algorithm `Reservoir`, and uniform allocation based SRS `Uniform`. Each method is given the same total memory of M records. The reservoir sampling algorithm maintains a uniform random sample of size M chosen without replacement from all the data seen so far - we expect the number of samples allocated to stratum i by `Reservoir` to be proportional to n_i, the number of points in the stratum. `Uniform` allocates the same amount of memory to each stratum that has been observed so far. If a stratum has too few data points to fill its current allocation, then the remaining memory is allocated uniformly among other strata, and this memory leftover redistribution may happen further, recursively.

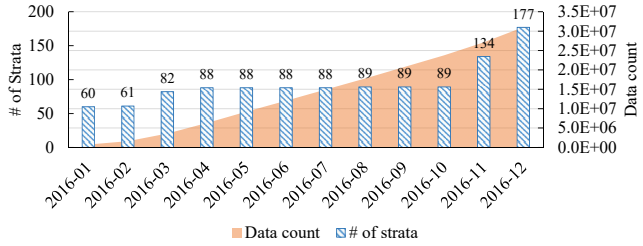
We implemented offline sampling methods, in which when a sample is desired, they use two passes through the data. One pass is to determine strata characteristics from which the sample size of each stratum is derived, and the other pass is to compute the samples. In the second pass, some strata may have fewer elements than the space allocated. All elements of those strata will be selected. We implemented the following offline sampling methods: `VOILA`, implemented as described in the paper; `Neyman`, that allocates the memory as Neyman's allocation; and `Neyman+`, an extended version of



(a) Relative Frequencies of Different Strata, during different months. The x-axis is the fraction of points observed so far. At different points in time, the relative (cumulative) frequency of each stratum is shown.

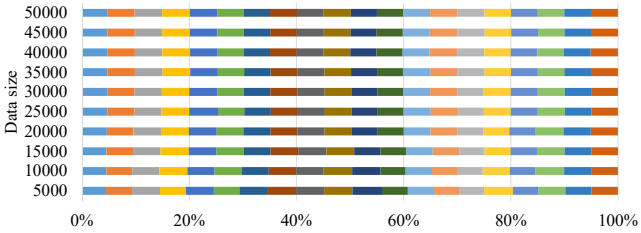


(b) Relative Standard Deviations of Different Strata, demonstrated by normalized cumulative standard deviations observed by the end of each month.

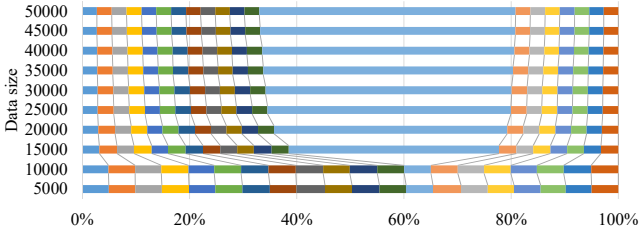


(c) The Number of Strata seen so far, and the number of data records, as a function of time.

Figure 2: Characteristics of the OpenAQ dataset.



(a) Relative Frequencies of Different Strata.



(b) Relative Standard Deviations of Different Strata

Figure 3: The Change in Data Characteristics over time of Synthetic dataset.

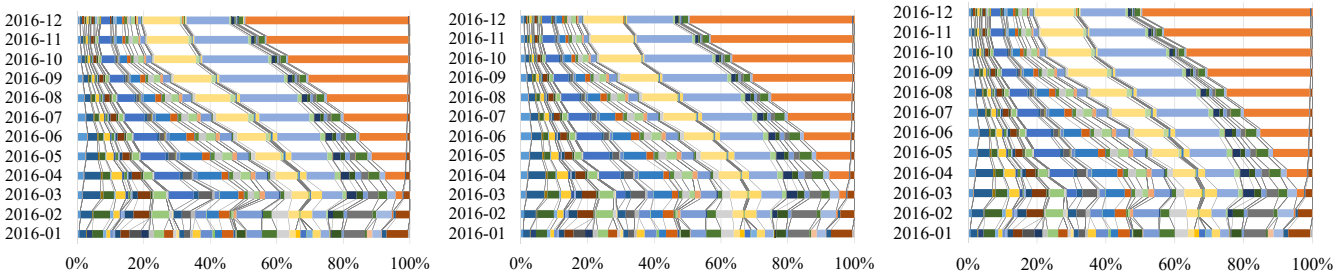
`Neyman`, that uses up to the maximum memory budget. As we have explained in previous sections that the original Neyman's allocation is non-optimal due to the lack of elements in the bounded strata. Since `Neyman` may leave portion of memory unused, for the sake of fairness in comparing different sampling methods using the same memory budget, we introduce `Neyman+` that redistributes the remaining memory from bounded strata equally to the rest of stata, which still have elements to be selected. The additional memory allocated may make some strata become bounded. Thus, we repeat the procedure recursively till either we make use of all the leftover memory or the sample database covers all the records we have. We compare `VOILA` to both `Neyman` and `Neyman+`, to show that `VOILA` not only overcomes the problem of bounded strata but also reuse the remaining unused memory in the optimal way.

6.2 Data

We used two datasets. The first is OpenAQ dataset [27], that contains more than 31 million records of air quality measurements from 7,923 locations in 62 countries around the world in 2016. The measurements includes particulate matter (PM10 and PM2.5), sulfur dioxide (SO_2), carbon monoxide (CO), nitrogen dioxide (NO_2), ozone (O_3), and black carbon (BC). The dataset is replayed in time order, to generate the data stream. Data is stratified based on the country of origin and the type of measurement, e.g., all measurements of carbon monoxide in the USA belong to one stratum, all records of sulphur dioxide in India belong to another stratum, and so on. The total number of strata at the end of observation is 177, as shown in Figure 2c.

We note that each stratum begins with zero record, and in the initial stages, each stratum has only a few points within – hence each stratum starts out as a bounded stratum. As more data are observed, many of the strata are not bounded anymore, but it is still the case that there are some strata with very few observations, when compared with other strata. Further, new strata are added as more data are observed, and more sensors are incorporated into the data stream. Figure 2c shows the number of strata, that have been observed at the end of each month, is increasing with more data. Figure 2a and 2b respectively show the cumulative frequency and standard deviation of the data over time. As seen, the relative frequency and relative standard deviation of different strata change significantly. As a result, the variance-optimal sample-size allocations to strata also change over time, and the streaming algorithms need to adapt to these changes.

In order to evaluate our proposed approaches on data for which we have more control, we created a synthetic data source, that generates streaming data. Each record i is a tuple $\langle sid, val \rangle$ where sid is the id of the stratum that record belongs to, and the value val. The number of strata is set to 20. Frequencies are equal between strata, *i.e.,* at any time, each stratum has approximately same amount of records. For a stratum j, the value of each record is drawn at random from Gaussian distribution with two parameters mean $\mu_j = 1$ and standard deviation σ_j, which is used to control the relative standard deviation among strata. For the first 10,000 records, we set $\sigma_j = 1$ for all the strata. After that, we change the standard deviation of a single arbitrarily se-



(a) Allocation due to `VOILA`

(b) Allocation due to `S-VOILA` with Single Element Processing

(c) Allocation due to `S-VOILA` with MiniBatch Processing (batch size = one-day of data)

Figure 4: Allocation due to different algorithms over time, OpenAQ data

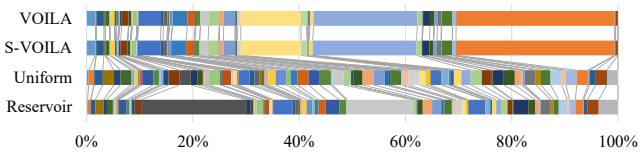


Figure 5: Allocation of sample sizes among strata after 9 months, OpenAQ data

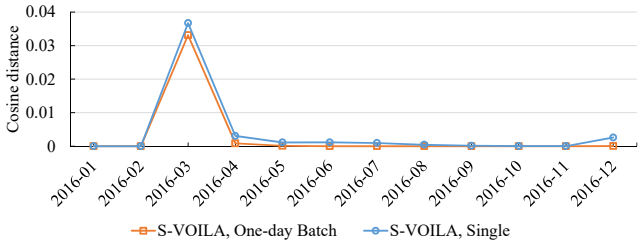


Figure 6: Cosine distance between the allocations due to `S-VOILA`, with Single and Minibatch Processing, and `VOILA`, OpenAQ data.

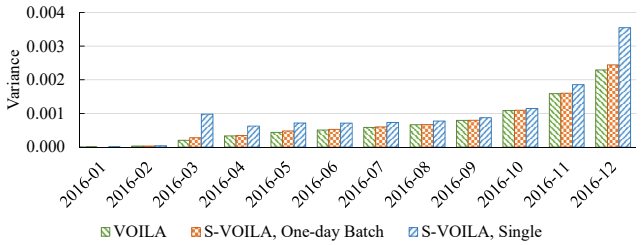


Figure 7: Variance of streaming `S-VOILA`, with Single and Minibatch Processing, compared with offline `VOILA`. Sample size is set to 1M records, for each method, OpenAQ data.

lected stratum (12) by setting $\sigma_{12} = 20$, while the other strata do not change. Figures 3a and 3b shows the relative frequencies and standard deviations of the synthetic dataset over time. While the frequencies are stable, the accumulated standard deviation shows how the stratum 12 changes.

The synthetic dataset allows us to capture the algorithm's behavior adapting to the change of data. In the real dataset, the characteristics of strata are changing frequently and continuously, so the allocation is result of the combined adaptation of multiple dynamic changes. With the synthetic dataset, which have a controllable single change, we can observe how the allocation affected by the change as well as how well the algorithm adapts to it to make the allocation converge to optimal again.

6.3 Allocations to Different Strata

We measured the allocation of samples to different strata. At any point in time (after M records have been observed), the total space taken by the sample is equal to M. Unless otherwise specified, the sample size M is set to 1 million records. The allocations to different strata can be seen as a vector of numbers that sum up to M (or equivalently, this can be normalized to sum up to 1), and we observe how this vector changes as more data arrive. Figure 5 shows the allocations at a single point in time, at the end of September 2016, for the OpenAQ data. From this figure, we see that the allocation of `S-VOILA` tracks that of the variance-optimal offline algorithm `VOILA` quite closely. As expected, `Reservoir`'s allocation is proportional to the volume of the stratum, while `Uniform`'s allocation is equal to all strata.

Figures 4a, 4b and 4c show the allocations over time produced by `VOILA`, `S-VOILA` with single element processing, and `S-VOILA` with minibatch processing where a minibatch contains data collected in each day. Visually, the allocations produced by the three methods track each other over time, showing that the streaming methods follow the allocation of the optimal offline algorithm, `VOILA`. To understand the difference between the allocations due to `VOILA` and `S-VOILA` quantitatively, we measure the cosine distance between the allocation vectors from `VOILA` and from `S-VOILA`. Figure 6 shows the average of 5 runs. As seen, the allocation vectors due to `S-VOILA` are highly similar to the vectors due to `VOILA`, where the cosine distance is close to 0 most of the time and less than 0.04 at all times. `S-VOILA` with minibatch processing yields an allocation that is closer to `VOILA` than `S-VOILA` with single element processing.

Comparison of Variance: Next we compared the variance of the estimates (Equation 1) from the stratified random samples produced by different stratified random sampling algorithms, offline or streaming. The result is shown in Figure 7. Generally, the variance of the sample due to each method increases over time, since the volume of data as well as the number of strata increase, while the sample size is fixed. The variance of `S-VOILA` using minibatch processing is very close to that of `VOILA`, showing that it is nearly variance-optimal at all times. The variance due to `S-VOILA`

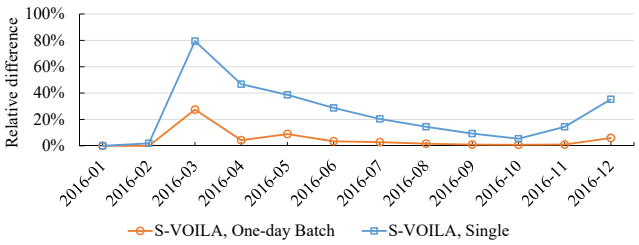


Figure 8: Relative difference of the variance of S-VOILA, with Single and Minibatch Processing, compared with the optimal variance due to VOILA, OpenAQ data.

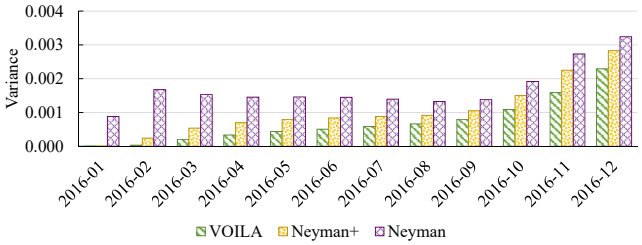


Figure 9: Variance of VOILA compares to Neyman and Neyman+ with equal sample size of 1M records, OpenAQ data.

with single element processing is higher.

Figure 8 shows the relative difference between the variance produced by a streaming algorithm ($\hat{x}$) and the optimal variance due to VOILA (x), defined as $\frac{\hat{x}-x}{x}$. Each point is the average of 5 runs. We note that the variance of both variants of S-VOILA are nearly equal to that of VOILA until March, when they start increasing relative to VOILA, and then converge back.

From analyzing the underlying data stream, we see that March is the time when a number of new strata appear in the data (Figure 2c), causing significant changes in the optimal allocation of samples to strata (this can also be seen in Figure 6 showing the cosine distance between the allocations). An offline algorithm such as VOILA can resample more elements from a stratum, if necessary, since it has access to all data from the stratum. However, a streaming algorithm such as S-VOILA cannot do so and must wait for enough new elements to arrive in these strata before it can "catch up" to the allocation of VOILA. Hence, S-VOILA with single element as well as with minibatch processing start showing an increase in the variance at such a point. When data become stable again, and more data arrive, the relative performance of S-VOILA improves. S-VOILA with minibatch processing approaches the optimal variance faster than S-VOILA with single element processing, which is as expected, since as the size of the minibatch increases, better optimization decisions are made with respect to which elements to exclude from the sample. In November and December, new strata appear again, and the relative performance is again affected. Overall, we note that S-VOILA with minibatch processing produces variance that is significantly closer to VOILA than S-VOILA with single element processing.

Among offline algorithms, we observe that Neyman produces variance that is higher than VOILA. While Neyman is known to be variance-optimal for unbounded strata, it is clearly not variance-optimal for bounded strata and its variance can often be far from that of VOILA, as seen in Figure 9. To show that VOILA not only addresses the problem of Neyman that leaves potion of memory to be unused, but also be optimal in allocating that remaining memory, we compared VOILA with Neyman+, the extended version of Neyman that reallocate the unused memory equally to the rest of unbounded strata. Neyman+ makes a fair comparison with VOILA since it uses up to the memory budget. Figure 9 shows that Neyman+ improves the original Neyman. Making uses of the remaining memory, Neyman+ provides a lower variance than Neyman. However, it reallocates the memory in a naive way while VOILA does it delicately, thus VOILA overperforms both Neyman and Neyman+.

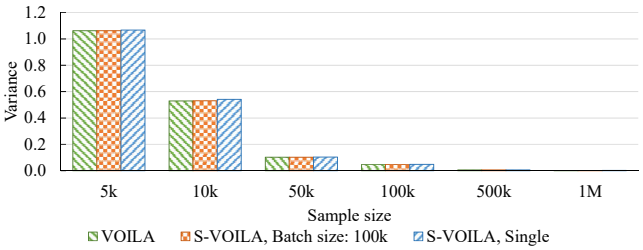


Figure 10: Impact of Sample Size on Variance, in September, OpenAQ data.

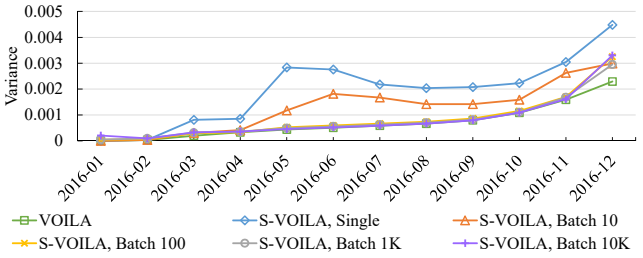


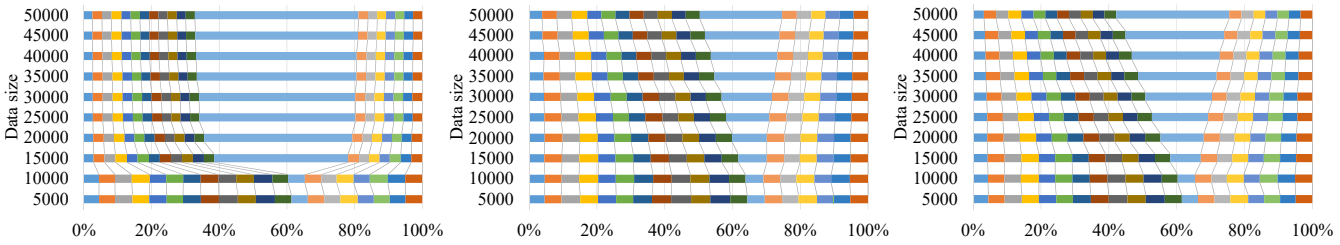
Figure 11: Impact of Batch Size on Variance. OpenAQ data.

Impact of Sample Size: To understand the sensitivity to the size of the sample, we conducted an experiment varying the sample size from 5000 up to 1 million records. We fixed the batch size to 100 thousand records. Figure 10 shows the snapshot in September 2016 of variances as a function of the sample size. For both VOILA and S-VOILA, with single element and minibatch processing, the variance decreases when the sample size increases. This is as expected, since larger samples produces better estimates of the population mean.

Impact of Batch Size: It is clear from Figure 7 that the variance of minibatch S-VOILA, where each batch contains data collected in a day, is performing significantly better than single element S-VOILA. In order to understand the impact of the batch size, we conducted an experiment where we tried different batch sizes for minibatch streaming S-VOILA, chosen from $\{1, 10, 10^2, 10^3, 10^4\}$. The results are shown in Figure 11. A batch size of 10 elements yields significantly better results than single element S-VOILA. A batch size of 100 or greater makes the variance nearly equal to the optimal variance.

6.4 Reaction to a Sudden Change in the Data Distribution

In a real-world dataset, such as OpenAQ, the allocation is



(a) Allocation due to `VOILA` across different strata.

(b) Allocation due to `S-VOILA` with Single Element processing.

(c) Allocation due to `S-VOILA` with Minibatch processing (batch size = 100).

Figure 12: The Change in allocations of different algorithms over time with synthetic dataset.

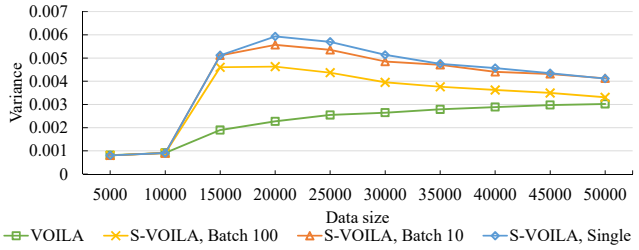


Figure 13: The variance changes due to a sole change in synthetic data.

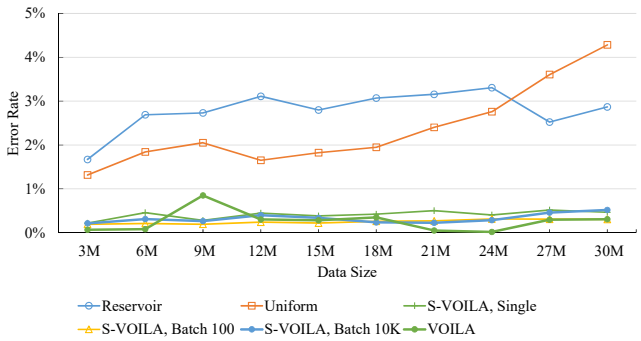


Figure 14: Query Performance as data size varies, with sample size fixed at 100,000. OpenAQ data.

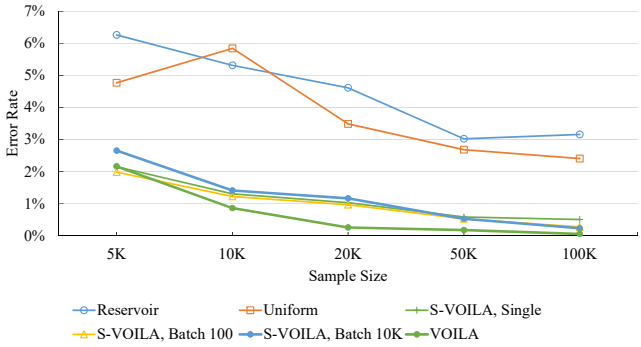


Figure 15: Query Performance as sample size varies, with data size fixed at 21 million. OpenAQ data.

affected by the combination of multiple factors that continuously change. To better observe the behavior of our algorithms under a single change, we conducted an experiment with our synthetic data. Figure 3b shows a single change in stratum 12, where the standard deviation suddenly increases from 1 to 20 after the first 10,000 records are generated. Meanwhile, the standard deviation of all the other strata are stable and their frequencies are stable as well. After this change, we will expect Stratum 12 to be given a greater sample size than the other strata. The memory budget is set to 1,000 records, which is 2% of the data size in the end of the experiment.

Figures 12a, 12b, and 12c show the allocations produced by `VOILA`, single element `S-VOILA`, and minibatch `S-VOILA`, respectively. As seen, `S-VOILA` slowly captures the sudden change in the data by giving Stratum 12 more sample space over time. `VOILA` is more sensitive to the change, due to the fact that `VOILA` works in an offline manner and is able to sample more data into Stratum 12 right after the change. Visually, minibatch `S-VOILA` is closer to the `VOILA` than single element `S-VOILA`.

Figure 13 shows the variance of different methods on synthetic data. At first, when the data is stable, all methods have nearly optimal variance. After a single change at 10,000 records, the variance of `VOILA` increases steadily, while those of different versions of `S-VOILA` increase at a faster rate. `S-VOILA` with a higher minibatch size has a lower variance. Interestingly, the variance of all versions of `S-VOILA` converge to that of the optimal method, `VOILA`, though `S-VOILA` with a minibatch of 100 elements converges the fastest.

6.5 Query Performance

We now evaluate the quality of these samples indirectly, through their use in approximate query processing, which is one of the major applications of sampling. The setup used is shown in Figure 1. The streaming sampler continuously maintains a stratified random sample of data (stored in memory), and use this sample to approximately answer aggregate queries, which are issued by the client. The offline sampler constructs its sample when needed, using `VOILA`, which takes two passes through the data. For evaluating the approximation error in query processing, we also implement an exact method for query processing, `Exact`, that stores every record in a table (stored in a MySQL database [31]) and answers a query using this table. While the exact method has zero error, its processing time is high, and so is its space overhead. Identical queries are made at the same time points in the stream to the different streaming and offline samplers, as well as to the exact query processor.

We measure the performance of the following samplers: `Reservoir`, `Uniform`, `S-VOILA`, `VOILA`, and `Exact`. We use

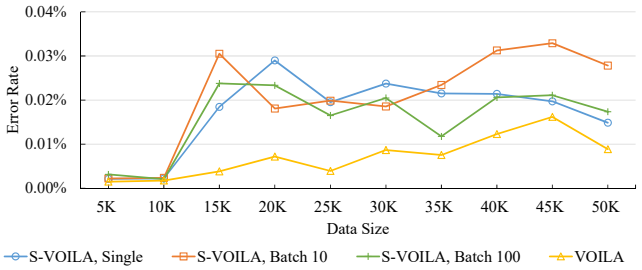


Figure 16: Query Performance on synthetic data as size of streaming data increases, with sample size fixed at 1,000 and one stratum's distribution changed at 10, 000.

the metric of relative error between the approximate answer and the exact answer, where the query asks for the mean of the data received across all strata. The sample size is set to 100, 000 for all samplers. For `S-VOILA`, we set minibatch size to be 1, 100, and 10, 000. Each data point is the mean of nine repetitions of the experiment with the same configuration.

Figure 14 shows the relative error as the size of the streaming data increases, while the sample size is held fixed. The query was executed every three million element arrivals, up to thirty million, which covers the entire year of 2016 in the OpenAQ dataset. We note that the relative performance between different methods remains similar for most data sizes. `Reservoir` has a consistent errors since it is mainly affected by sample size rather than data size. `Uniform` is affected by total number of strata and as expected, we see an increasing error when the data size reaches 24 million, where the number of strata increases suddenly as shown in Figure 2c, November 2016. The performance of `VOILA` and `S-VOILA` increase slightly with data size, though at much lower rates than `Reservoir` and `Uniform`. We note that `S-VOILA` with any minibatch size is very close to `VOILA`.

Figure 15 shows the impact of the sample size, as it varies from 5,000 to 100,000, and the queries were executed at a fixed time of stream to see how sample size would affect the accuracy of answering queries. As expected, all methods benefit from increased sample size. We observed `S-VOILA` and `VOILA` perform significantly better than `Reservoir` and `Uniform` even with smaller sample sizes. Another observation of `S-VOILA` is that a larger minibatch size does not always guarantee better accuracy. When total sample size is small, each stratum is allocated with a smaller space and there are fewer bounded strata. Therefore, the eviction made by single and minibatch processing affected the performance less. With our configuration, `S-VOILA` with minibatch ten thousand elements did not yield a better accuracy until sample size was set to one hundred thousand.

We also test the query performance of `S-VOILA` and `VOILA` on synthetic data, with the same configuration as in Section 6.4. Figure 16 shows the performance of a query across all strata. The first observation is `VOILA` is less affected by the distribution change since it samples from all the received data, while `S-VOILA` methods had to discard data on the fly. Another observation is that `S-VOILA` with a larger minibatch size usually yields better performance.

7. REFERENCES

[1] S. Acharya, P. Gibbons, and V. Poosala. Congressional samples for approximate answering of group-by queries. In *Proc. SIGMOD*, pages 487–498, 2000.

[2] S. Acharya, P. B. Gibbons, V. Poosala, and S. Ramaswamy. The aqua approximate query answering system. In *Proc. SIGMOD*, pages 574–576, 1999.

[3] S. Agarwal, B. Mozafari, A. Panda, H. Milner, S. Madden, and I. Stoica. BlinkDB: Queries with bounded errors and bounded response times on very large data. In *Proc. EuroSys*, pages 29–42, 2013.

[4] M. Al-Kateb, B. S. Lee, and X. S. Wang. Adaptive-size reservoir sampling over data streams. In *Proc. SSDBM*, page 22, 2007.

[5] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. Models and issues in data stream systems. In *Proc. PODS*, pages 1–16, 2002.

[6] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. In *Proc. SIGMOD*, pages 539–550, 2003.

[7] B. Babcock, M. Datar, and R. Motwani. Sampling from a moving window over streaming data. In *Proc. SODA*, pages 633–634, 2002.

[8] V. Braverman, R. Ostrovsky, and G. Vorsanger. Weighted sampling without replacement from data streams. *Inf. Process. Lett.*, 115(12):923–926, 2015.

[9] V. Braverman, R. Ostrovsky, and C. Zaniolo. Optimal sampling from sliding windows. In *Proc. PODS*, pages 147–156, 2009.

[10] S. Chaudhuri, G. Das, and V. Narasayya. Optimized stratified sampling for approximate query processing. *ACM Trans. Database Syst.*, 32(2), 2007.

[11] W. G. Cochran. *Sampling Techniques*. John Wiley & Sons, New York, third edition, 1977.

[12] G. Cormode, V. Shkapenyuk, D. Srivastava, and B. Xu. Forward decay: A practical time decay model for streaming systems. In *Proc. ICDE*, pages 138–149, 2009.

[13] G. Cormode, S. Tirthapura, and B. Xu. Time-decaying sketches for robust aggregation of sensor data. *SIAM J. Comput.*, 39(4):1309–1339, 2009.

[14] P. S. Efraimidis and P. G. Spirakis. Weighted random sampling with a reservoir. *Inf. Process. Lett.*, 97(5):181–185, 2006.

[15] R. Gemulla and W. Lehner. Sampling time-based sliding windows in bounded space. In *Proc. SIGMOD*, pages 379–392, 2008.

[16] R. Gemulla, W. Lehner, and P. J. Haas. Maintaining bounded-size sample synopses of evolving datasets. *The VLDB Journal*, 17(2):173–201, 2008.

[17] P. B. Gibbons and S. Tirthapura. Estimating simple functions on the union of data streams. In *Proc. SPAA*, pages 281–291, 2001.

[18] P. J. Haas. Data-stream sampling: Basic techniques and results. In *Data Stream Management*, pages 13–44. Springer, 2016.

[19] T. Johnson and V. Shkapenyuk. Data stream warehousing in tidalrace. In *Proc. CIDR*, 2015.

[20] S. Joshi and C. Jermaine. Robust stratified sampling plans for low selectivity queries. In *Proc. ICDE*, pages 199–208, 2008.

[21] P. Kranen, S. Günnemann, S. Fries, and T. Seidl. Mc-tree: Improving bayesian anytime classification. In *Proc. SSDBM*, pages 252–269, 2010.

[22] S. L. Lohr. *Sampling: Design and Analysis*. Duxbury Press, 2nd edition, 2009.
[23] I. Mcleod and D. Bellhouse. A convenient algorithm for drawing a simple random sample. *Journal of the Royal Statistical Society. Series C. Applied Statistics*, 32:182–184, 1983.
[24] X. Meng. Scalable simple random sampling and stratified sampling. In *Proc. ICML*, pages 531–539, 2013.
[25] J. Neyman. On the two different aspects of the representative method: The method of stratified sampling and the method of purposive selection. *Journal of the Royal Statistical Society*, 97(4):558–625, 1934.
[26] T. D. Nguyen, M.-H. Shih, D. Srivastava, S. Tirthapura, and B. Xu. Variance-optimal offline and streaming stratified random sampling. (arXiv link here).
[27] http://openaq.org.
[28] S. K. Thompson. *Sampling*. Wiley, 3rd edition, 2012.
[29] Y. Tillé. *Sampling Algorithms*. Springer-Verlag, 1st edition, 2006.
[30] J. S. Vitter. Optimum algorithms for two random sampling problems. In *Proc. FOCS*, pages 65–75, 1983.
[31] M. Widenius and D. Axmark. *Mysql Reference Manual*. O'Reilly & Associates, Inc., 1st edition, 2002.

APPENDIX

A. PROOFS FROM SECTION 3, SAMPLE SIZE REDUCTION

We present the details of algorithm `FastSSR`, which computes the variance-optimal sample size reduction in time $O(r \log r)$ where r is the number of strata.

Definition 2. *Let $Q[1..r]$ be an array of (x, y, z) tuples, where each $Q[i]$ is initialized as $(i, n_i\sigma_i, s_i/(n_i\sigma_i))$. Array Q is then sorted on its z dimension.*

Lemma 4. *Under any given memory budget M, if there exists at least one sample that is not oversized, then the collection of the identifiers of the samples that are not oversized must occupy a continuous prefix of the array Q.*

Proof. Recall that under a memory budget M, the Neyman allocation size for stratum i is $M_i = n_i\sigma_i/D$, where $D = \sum_{i=1}^{r} n_j\sigma_j$. A sample S_i is not oversized if and only if $s_i \leq M_i$, i.e., $s_i/(n_i\sigma_i) \leq 1/D$. A sample S_i is oversized if and only if $s_i > M_i$, i.e., $s_i/(n_i\sigma_i) > 1/D$. Because array Q is in the ascending order of its z dimension, the lemma is proved. □

Lemma 4 implies that we can linearly walk along the array Q from $Q[1]$ toward $Q[r]$. By comparing the sample size and the Neyman allocation size for each stratum we are looking at during the walk, we will be able to find the collection of samples that are not oversized, under the new target memory budget M'.

After finding the prefix of the Q array that represents the collection of samples that are not oversized, we pause the walk and then set the new memory M' budget to be M' minus the total size of the samples in the prefix. Then, we treat the remaining part (after excluding the prefix) of the array Q as the current array Q and do the same walk under the new memory budget M'.

The walk will stop if we do not see any sample that is not oversized under the current memory budget M'. In that case, we just set the size of the samples in the current array Q to be their Neyman allocation size, under the current memory budget.

In order to avoid the recomputation of D, which is needed in computing the Neyman allocation, for every new memory budget during the walk, we precompute the D for every suffix of the array Q and save the result in the y dimension of the Q array.

The method `FastSSR` in Algorithm 6 shows the pseudocode of this faster algorithm for variance-optimal sample size reduction.

We now repeat the statement of Theorem 2, before presenting its proof.

(1) The `FastSSR` *procedure in Algorithm 6 finds the correct size of each sample of a stratified random sample, whose memory budget is reduced to M, such that the increase of the variance V is minimized. (2) The worst-case time cost of* `FastSSR` *on a stratified random sample across r strata is $O(r \log r)$.*

Proof. (1) The correctness of the procedure follows from Lemmas 2–3, Observation 1, and Lemma 4. (2) The time cost of `FastSSR` is dominated by the step of sorting array Q on its z dimension (Line 4), so the worst-case time cost of `FastSSR` is $O(r \log r)$. □

Algorithm 6: `FastSSR`(M): A fast implementation of Sample Size Reduction without using recursion.

Input: The strata under consideration is implicitly $\{1, 2, \ldots, r\}$. M is the target total sample size.
Output: For $1 \leq i \leq r$, $\mathcal{L}[i]$ is set to the final size of sample for stratum i, such that the increase of the variance V is minimized.

```
1  Allocate L[1..r], an array of numbers
2  Allocate Q[1..r], an array of (x, y, z) tuples
3  for i = 1 ... r do  Q[i] ← (i, n_i σ_i, s_i/(n_i σ_i));
4  Sort array Q in ascending order on the z dimension
5  for i = (r − 1) down to 1 do
6      Q[i].y ← Q[i].y + Q[i + 1].y

7  M_new ← M; D ← Q[1].y
8  for i = 1 ... r do
9      M_{Q[i].x} ← M · n_{Q[i].x} σ_{Q[i].x} / D
10     if s_{Q[i].x} > M_{Q[i].x} then break
11     L[Q[i].x] ← s_{Q[i].x}
12     M_new ← M_new − s_{Q[i].x}

       // Check the next sample, which must exist.
13     M_{Q[i+1].x} ← M · n_{Q[i+1].x} σ_{Q[i+1].x} / D
14     if s_{Q[i+1].x} > M_{Q[i+1].x} then // oversized
15         M ← M_new; D ← Q[i + 1].y

   // Reduce sample size to target.
16 for j = i..r do
       // Desired size for S_{Q[j].x}
17     L[Q[j].x] ← M · n_{Q[j].x} σ_{Q[j].x} / D
18 return L
```

B. PROOFS FROM SECTION 5, STREAMING SRS

We now present the proof of Theorem 4, a lower bound on the variance of any streaming algorithm for SRS. The statement of the theorem is as follows:

Any stratified random sample maintained over a stream across r stratra must have a multiplicative error of at least $\Omega(r)$ in the worst case.

Proof. We use proof by contradiction. Suppose that it is possible to maintain an approximate stratified random sample with a multiplicative error less than r.

Consider an input stream where the ith stratum consists of elements in the range $[i, i+1)$, where the right endpoint of the stratum does not include $i+1$. Suppose the stream so far has the following elements. For each i from 1 to r, there are $(\alpha-1)$ copies of element i and one copy of $(i+\varepsilon)$ where $0<\varepsilon<1$ and $\alpha \geq 3$. After observing these elements, for each stratum i, $1 \leq i \leq r$, we have:

$$n_i = \alpha, \qquad \mu_i = i + \frac{\varepsilon}{\alpha},$$

$$\sigma_i = \sqrt{\left((\alpha-1)\left(\frac{\varepsilon}{\alpha}\right)^2 + \left(\varepsilon - \frac{\varepsilon}{\alpha}\right)^2\right)\Big/\alpha} = \frac{\sqrt{\alpha-1}}{\alpha}\varepsilon.$$

Observe that, due to the memory budget M, at least one stratum has its sample size no more than M/r. Without loss of generality, let's say that stratum is stratum 1.

Suppose an element of value $(2-\varepsilon)$ arrives in the stream, where $\varepsilon = 1/(r-1)$. This element belongs to stratum 1. Let n_1', μ_1', and σ_1' denote the new size, mean, and standard deviation of stratum 1 after this element arrives.

$$n_1' = \alpha + 1, \qquad \mu_1' = 1 + \frac{1}{\alpha+1},$$

$$\begin{aligned}\sigma_1' &= \sqrt{\frac{(\alpha-1)\left(\frac{1}{\alpha+1}\right)^2 + \left(\varepsilon - \frac{1}{\alpha+1}\right)^2 + \left(1-\varepsilon-\frac{1}{\alpha+1}\right)^2}{\alpha+1}} \\ &= \sqrt{\frac{\varepsilon^2 + (1-\varepsilon)^2 - \frac{1}{\alpha+1}}{\alpha+1}}.\end{aligned}$$

It follows that:

$$(\alpha+1)\sqrt{\frac{\frac{1}{2}-\frac{1}{\alpha+1}}{\alpha+1}} \leq n_1'\sigma_1' \leq (\alpha+1)\sqrt{\frac{1-\frac{1}{\alpha+1}}{\alpha+1}} \qquad (11)$$

$$\Longrightarrow \frac{\sqrt{\alpha}}{2} \leq n_1'\sigma_1' \leq \sqrt{\alpha} \qquad \text{(Note: } \alpha > 2\text{)} \qquad (12)$$

In 11, the left inequality stands when $\varepsilon = 1/2$ and the right inequality stands when $\varepsilon = 0$ or 1. We also have:

$$\sum_{i=2}^{r} n_i\sigma_i = (r-1)\alpha\frac{\sqrt{\alpha-1}}{\alpha}\varepsilon = \sqrt{\alpha-1} \qquad \left(\text{Note: } \varepsilon = \frac{1}{r-1}\right)$$

$$\Longrightarrow \frac{\sqrt{\alpha}}{2} \leq \sum_{i=2}^{r} n_i\sigma_i \leq \sqrt{\alpha} \qquad \text{(Note: } \alpha > 2\text{)} \qquad (13)$$

Let V denote the sample variance of the stratified random sample maintained over the stream of $(r\alpha+1)$ elements. Let V^* denote the smallest sample variance that one can get from a stratified random sample from these $(r\alpha+1)$ date elements. Let $\Delta = \left(n_1'\sigma_1'^2 + \sum_{i=2}^{r} n_i\sigma_i^2\right)/n^2$.

We observe the facts that (1) after processing these $(r\alpha+1)$ elements, the sample size $s_1 \leq M/r+1$. (2) The portion of the sample variance contributed by strata $2, 3, \ldots, r$ is minimized if the memory budget for these strata, which is no more than M, are equally shared, because all $n_i\sigma_i$ are equal for $i = 2, 3, \ldots, r$. Using these two facts and the definition of the sample variance in equation 1, we have:

$$\begin{aligned}V &= \frac{1}{n^2}\left(\frac{n_1'^2\sigma_1'^2}{s_1} + \sum_{i=2}^{r}\frac{n_i^2\sigma_i^2}{s_i}\right) - \Delta \\ &\geq \frac{1}{n^2}\left(\frac{n_1'^2\sigma_1'^2}{M/r+1} + \sum_{i=2}^{r}\frac{n_i^2\sigma_i^2}{M/(r-1)}\right) - \Delta \\ &\geq \frac{1}{n^2}\left(\frac{\alpha/4}{M/r+1} + \sum_{i=2}^{r}\frac{(\alpha-1)\varepsilon^2}{M/(r-1)}\right) - \Delta \\ &= \frac{1}{n^2}\left(\frac{\alpha/4}{M/r+1} + \frac{\alpha-1}{M}\right) - \Delta \qquad \left(\text{Note: } \varepsilon = \frac{1}{r-1}\right)\end{aligned}$$

On the other hand, the smallest sample variance V^* is achieved by using the Neyman allocation of the memory budget M, assuming each stratum has sufficient data to fill its sample size assigned by the Neyman allocation. By Inequalities 12 and 13, we know that in the Neyman allocation for the current stream of $r\alpha+1$ elements, stratum 1 uses at least $M/3$ memory space, whereas all other strata equally share at least $M/3$ memory space as well because all $n_i\sigma_i$ are equal for $i = 2, 3, \ldots, r$. Using these observations into Equation 1, we have:

$$\begin{aligned}V^* &\leq \frac{1}{n^2}\left(\frac{n_1'^2\sigma_1'^2}{M/3} + \sum_{i=2}^{r}\frac{n_i^2\sigma_i^2}{M/3(r-1)}\right) - \Delta \\ &\leq \frac{1}{n^2}\left(\frac{\alpha}{M/3} + \sum_{i=2}^{r}\frac{(\alpha-1)\varepsilon^2}{M/3(r-1)}\right) - \Delta \\ &= \frac{1}{n^2}\frac{6\alpha-3}{M} - \Delta \qquad \left(\text{Note: } \varepsilon = \frac{1}{r-1}\right)\end{aligned}$$

Because $\Delta \geq 0$ and $M > r$, we have:

$$\frac{V}{V^*} \geq \frac{V+\Delta}{V^*+\Delta} = \Omega(r)$$

□